SMART BUILDINGS – A CONCEPT FOR AD-HOC CREATION AND REFINEMENT OF 3D BUILDING MODELS

J. Döllner, H. Buchholz, F. Brodersen, T. Glander, S. Jütterschenke, A. Klimetschek

University of Potsdam, Hasso-Plattner-Institut, 14482 Potsdam, Germany
(juergen.doellner, henrik.buchholz, tassilo.glander, sascha.juetterschenke, alexander.klimetschek)@hpi.uni-potsdam.de
florian.brodersen@t-systems.com

KEY WORDS: Architecture, Cartography, Modeling, Reconstruction, Visualization, Building, City, Virtual Reality

ABSTRACT:

This paper presents smart buildings, a concept for ad-hoc creation and refinement of 3D building models. A smart building represents a building's geometry and appearance on a per-floor basis. A building's floor basically consists of a ground plan and walls, whereby each floor and its parts can be specified independently. In addition, smart buildings provide and maintain detailed building semantics and allow for attaching application-specific semantics not just to the whole building but also to specific floors and sections of façades. Smart buildings provide intuitive means for constructing, reshaping, and refining 3D building models. In particular, they provide an effective solution for integrating building models of different level-of-detail within a uniform framework. With smart buildings, authoring systems for 3D city models can implement cost-effective intuitive tools for the maintenance and incremental development of 3D city models. The concept can serve both as a schema for implementing 3D city model systems as well as provide a suggestion for further extensions to standards regarding 3D city models such as CityGML.

1. INTRODUCTION

1.1 Incremental Development of 3D City Models

A growing number of cities and communities are building up geodata infrastructures, which manage and distribute 2D and 3D geodata. 3D city models integrate 2D and 3D geodata using the paradigm of the virtual city. Important components of virtual 3D city models include models of buildings, transportation networks and systems, vegetation, and environment.

After a long time of feasibility studies, prototypic implementations, and geo-specific uses, first domain-specific applications and systems appear that incorporate virtual 3D city models as essential parts such as in facility management applications, real estate portals as well as entertainment and education products. Hence, we can expect a large number of potential users and uses that require customized and specialized versions of 3D city models.

The question arises how 3D city models can be maintained and how the process of authoring and customization can be defined. Consider the example where urban planners want to incorporate a new plan for the development of a district into an existing 3D city model. They have to rebuild and refine parts of the model. Therefore, 3D city models should facilitate the incremental development of their components.

1.2 Creating 3D City Models

The initial creation of 3D city models represents a technically challenging and economically cost-intensive task. Building models, the major components of 3D city models, can be systematically built based on a wide range of *techniques for acquisition, classification, and analysis of urban data* derived from, for example, laser scans, aerial photography, and cadastre information bases. For a detail overview of related methods, see Hu et al. (2003), Ribarsky et al. (2002), Bauer et al. (2002), and Förstner (1999).

The geometric level-of-detail of building models derived by *automatic techniques*, however, is relatively low such as in the case of block models or models with approximated roof geometry. Most applications require more geometric details, which can be either created using *semi-automatic techniques* of photogrammetry (e.g., Ulm 2003; Karner 2004) or manual modeling based on CAD or 3D modeling tools.

Complementary, *procedural techniques* for creating virtual 3D city models have emerged in the scope of computer graphics, intended for research, simulation, and educational purposes. In particular, specific markets such as the movie and game industry have a high demand for a cost and time-efficient creation of realistic, complex urban environments.

Parish and Müller (2001) present a system that creates a complete 3D city model using a small set of statistical and geographical input data. The system provides tools for generating roads, allotments, buildings, and procedural textures. Wonka et al. (2003) introduce a concept for instant architectural building models. In their approach, building designs are derived using parametric set grammars, an attribute matching system, and a separate control grammar to derive buildings having a large variety of different styles and design ideas.

1.3 City Model Representations

Independent of the way of creation, 3D buildings traditionally are exported as 3D scenes in standard 3D formats, e.g., VRML, X3D, or 3D Studio Max. While scene description languages and scene graph systems offer a broad repertoire of generic graphics functionality, they do not provide specialized means for 3D geodata-based objects. Consequently, it is generally difficult to represent and to take advantage of object semantics.

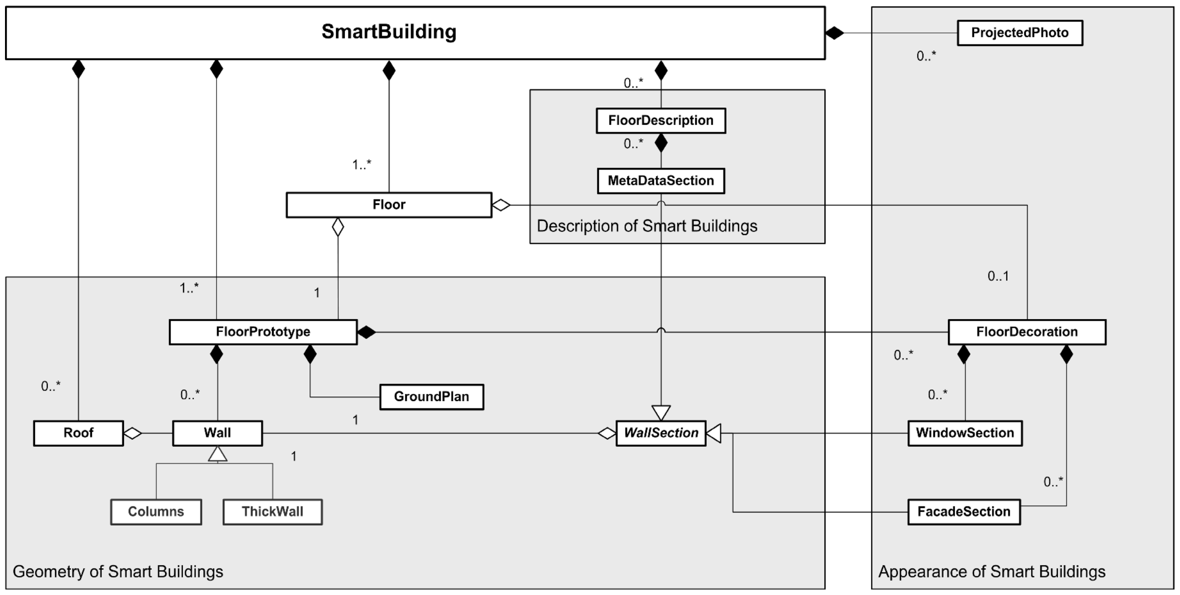


Figure 1: UML class diagram of the smart building concept.

The CityGML initiative (Kolbe et al. 2005) addresses the need for a domain-specific, semantics-preserving format for 3D city model components. CityGML supports different level-of-details of buildings (Altmaier and Kolbe 2003): block models (LOD-1), simplified buildings including roof geometry (LOD-2), detailed buildings (LOD-3), and detailed buildings including indoor models (LOD-4). These categories refer to principal *quality levels* but do not imply a specific kind of representation or encoding technique.

At the moment CityGML supports the four discrete levels of detail. However, an integral solution for a continuous modeling across these quality levels would enhance the expressivity, address practically important intermediate quality levels, and enable developing efficient cross-LOD rendering and interaction techniques. For example, in many application users want to smoothly transform an LOD-1 building into an LOD-4 building during a refinement process.

Conceptually, there is also no sharp (mathematical) distinction between LOD-2 and LOD-3 buildings: Even for LOD-2 buildings, it can be necessary to add significant geometric details such as an entrance hall if these details are perceptually important – the corresponding quality level could be considered to be somewhere in the range of 2 and 3. Similarly, in modern architecture there the distinction between indoor and outdoor is softened, e.g., if large parts of a facade are made of glass, a minimal indoor model including floors and main walls is required – the corresponding quality level would be somewhere between 3 and 4.

The concept of smart buildings extends the CityGML building model, providing a modeling schema for representing buildings at *continuous level-of-detail*. A smart building represents a building's geometry and its texture-based appearance on a per-floor basis. The scope of smart buildings encompasses simple block models, models with roof geometry, detailed geometric models, and architectural models including principal interior parts. Thereby, smart buildings facilitate the incremental development, and the ad-hoc refinement, that is, they enable a cost and time-effective management of 3D city models.

1.4 Real-Time Rendering of 3D City Models

The representation of building models should take into account that in most applications building models need to be displayed in real-time. Except for small 3D city models, an internal rendering-optimized representation will be required. In particular, each model must be decomposed into rendering primitives (e.g., triangle strips). Optimization strategies such as view-frustum culling, occlusion culling, and back-face culling (Akenine-Möller and Haines 2002; Schaufler 1998) operate on general graphics primitives. Out-of-core visualization techniques further improve the rendering process to cope with massive data sets accessed via external media (Davis et al. 1999; Lindstrom and Pascucci 2002). Specialized strategies exist for large-scale virtual environments such as described by Willmott et al. (2001).

Frequently, 3D models generated by architectural tools and systems are less suited for real-time rendering – the exported models have to be pre-processed first. The implementation of efficient rendering algorithms is simplified if a unified internal representation of buildings across all quality levels would be available. Smart buildings intend to supply such a framework.

2. SMART BUILDING REPRESENTATION

This section introduces the object-oriented model of smart buildings as illustrated by its UML class diagram in Figure 1. A smart building represents a single building entity of a 3D city model; it is implemented as a container object that aggregates floors, floor descriptions, and appearance information.

2.1 Floors

A *SmartBuilding* object is composed of one or more *Floor* objects, each of which defines the ground plan as well as the walls placed on top of it.

A *Floor* object always refers to a *FloorPrototype* object, which actually contains the floor specification. We introduce this indirection because the prototype concept compactly represents similar floors within a multi-storey building.

Each floor prototype is defined by its *GroundPlan* object. It consists of one or more polygons that define the potential area on which walls may be constructed. Each polygon is defined by its outer loop and optionally inner loops that model holes (e.g., courtyards). We allow for multiple polygons since a floor may consist of several components not necessarily directly linked.

A floor can also define roof geometry that is put onto the top of certain walls or onto the top of the whole floor. The roof geometry creaton is based on the straight-skeleton approach described by Felkel and Obdrmalek (1998) and implemented according to Laycock and Day (2003).

A ground plan defines the base plate for walls and, therefore, it is a mandatory object for each floor. It additionally defines its height, that is, the thickness. The thickness can be zero in the case of an abstract floor plane or can be positive if the floor should be modeled as 3D solid object. For example, with solid ground plans terraces or similar protrusion building elements can be directly expressed with an appropriate 3D geometry.

2.2 Walls

On top of the ground plan we can place *Wall* objects. A wall represents a vertical, planar, finite polygon that is constrained to directly lie on top of its ground plan. By default, a wall has a thickness of zero, that is, it is represented as a single polygon. Those walls are sufficient if they form a closed surface and can therefore be seen only from outside. A specialized wall object of type *ThickWall*, however, defines a positive thickness. Here, the wall is geometrically instantiated as a 3D solid object.

Walls are constrained to be non-intersecting in the same floor. If walls intersect, they have to be split into parts. Walls are not constrained with respect to their height, that is, a wall can have less height as the floor itself or can even be higher. For example, low walls can represent the fronts of a balcony, whereas the sides of a chimney starting at the basement floor can be extended above the roof.

It is possible to specify the walls of a floor independently of other floors, that is, we do not constrain the wall structures across floors to keep the degrees of freedom high. There is no validation with respect to the statics of a building since smart buildings are primarily intended for representing building geometry delivered by CAD systems and authoring tools that may provide this kind of validation.

With the two introduced building blocks, floors and walls, we are able to express frequently occurring geometric building characteristics such as protrusions, passages, terraces, penthouses, etc. This approach keeps the implementation as well as the usage of smart buildings simple.

2.3 Floor Decoration

FloorDecoration objects are responsible for specifying the appearance of smart buildings; they are parts of a floor prototype. The strategy of the floor decoration is to assign appearance information to sections of walls, identified by *WallSection* objects. These sections can refer to a whole wall or only to part of a wall.

The appearance of a wall section can be defined by two types of *WallSection* objects: *WindowSections* and *FacadeSections*. A *FacadeSection* describes the overall appearance of a wall. One way to define a façade section is the assignment of a façade pattern texture containing windows as well as the surrounding surface material as a single image. The second, more flexible way, is to model the windows explicitly. In this case, the *FacadeSection* describes only the wall material, and an additional *WindowSection* defines the positions and appearance of all visible windows separately.

2.4 Projective Textures

A smart building can specify projective textures to specify its appearance. A *ProjectedPhoto* object refers to a texture, e.g., a photo taken from the building's facades. This texture is projected into the 3D space using auxiliary "ghost-wall" that is independent of floors, ground plans, and walls.

Projective textures represent an orthogonal approach for providing appearance of smart buildings. While floor decorations allow for a procedural modeling of facades, projective textures are intended for image data captured from real-world buildings. Both approaches can be combined.

2.5 Application-Specific Building Data

Application-specific data can be assigned to individual parts of smart buildings using *FloorDescription* objects. Any number of descriptions can be associated with a smart building. A generic attribute table stores key-value pairs of information.

Specialized wall sections, *MetaDataSections*, are used to geo-reference application-specific data. Geo-referencing is important because it allows us to assign information to specific parts of a building. There is no restriction with respect to overlapping and multiple meta-data sections. For example, a smart building may define multiple company addresses, each of which is assigned to a different floor, and a general building description assigned to the whole facade.

3. EDITING SMART BUILDINGS

The concept of smart buildings concentrates on principal parts of a building including roofs, floors, and facades. This allows us to capture a large bandwidth of building types and facilitates the design of intuitive authoring tools for creating, manipulating, and refining smart buildings.

3.1 From Block Models to Smart Buildings

To illustrate a typical use case, assume that an initial block model should be refined. First, it is transformed into a simple smart building by splitting the model into a number of floors having the same ground plans and outside walls. Figure 2 shows the refinement process of a block building into a smart building:

1. The initial block model results from extruding 2D ground plans (Figure 2a).
2. The model is subdivided into floors according to a list of floor heights (Figure 2b).
3. The top floors are modified to distinguish different building parts, roof geometry is added to the top floor of the center building part, the basement floor is enhanced by columns, and a balcony is added to the left part (Figure 2c).
4. Façade textures are added, which can be specified either by composing different texture patterns or as projective textures, e.g., using digital photography (Figure 2d).

The example shows a frequent requirement in applications based on 3D city models: The existing 3D city model needs to be *partially and incrementally developed* according to current project goals or management decisions.

3.2 Smart Building Editor

For the design of the smart building editor we assume that non-expert users (e.g., non-architects) should be able to construct and refine smart buildings. Since floors are the dominant conceptual elements, the 2D floor editor is the core component.

Figure 3 shows a snapshot of the smart building editor of our implementation. The object tree (Figure 3 top-left) lists the *Floor* objects and indicates the corresponding floor prototypes. The selected floor prototype can be edited by the 2D editor widget (Figure 3 bottom-right). The user can directly manipulate ground plan polygons, walls, and wall sections. Changes are immediately reflected by the 3D view of the smart building.

Most frequent operations include adding and modifying floors, walls, and wall sections. For instance, it is easily possible to extend a smart building by a penthouse by replicating its top floor and reshaping the ground plan of the new floor. A selected floor may also be enhanced by adding geometric details such as columns. To integrate application data, e.g., user can specify a shop window-texture as a section of a wall. To edit a wall section, the user specifies the start and end points of the section on the ground plan polygon.

3.3 Editing Ground Plans and Walls

In the simplest case, the base plate of a floor is completely hidden by surrounding walls. If a building contains a terrace or if two subsequent floors have different ground plans, the base platform becomes partially visible (such as in Figure 2c and 2d).

Walls are specified by polylines and height values. Singular line segments of a polyline can be replaced by arcs to model curved shapes. Most walls are only visible from outside and, therefore, do not need to be solid. Freestanding walls, however, such as the boundaries of a terrace, can be seen also from above or from an indoor perspective. In this case, thick walls are used. The special case of *ColumnWalls* allows for the alignment of a row of columns along the wall's polyline. The height of walls can be automatically determined by the distance to the next floor or be specified explicitly.

Walls together with appropriate textures can be instrumentalized to model railing, cutouts, or interior decorations such as paintings. In general, the textures applied to such a wall will be mostly transparent. Although walls are planar objects, it provides a straightforward method to incorporate those elements for visualization and illustration purposes.

3.4 Editing Floor Decorations

Smart buildings support two techniques of façade texturing:

- Projective textures: These textures are orthogonally

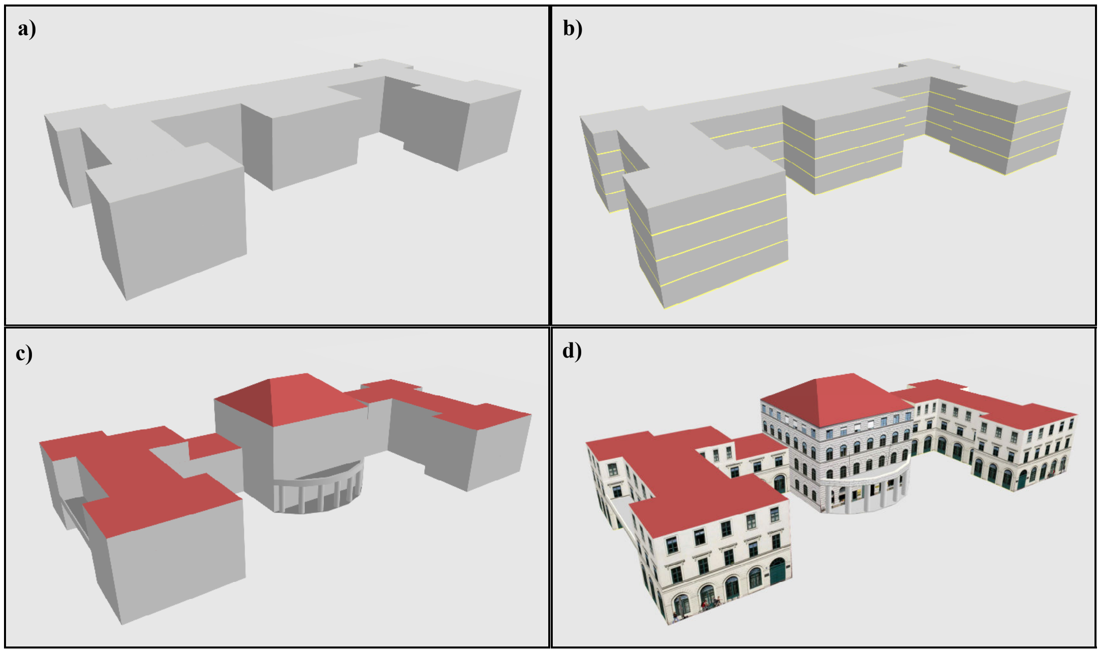
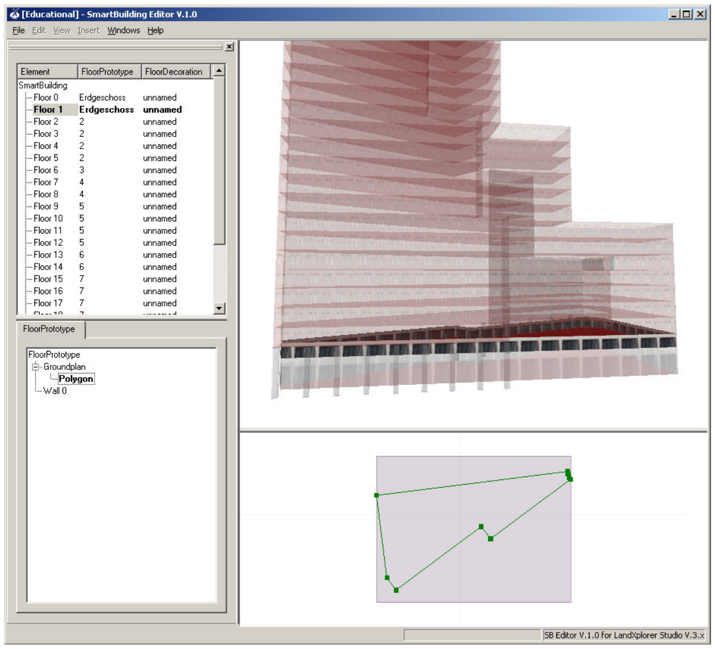


Figure 2: Transforming a block building into a smart building. a) Block building. b) Block building split into floors. c) Refined geometry. d) Added facade textures.


[Educational] - SmartBuilding Editor V.1.0
File
Edit
View
Insert
Windows
Help
FloorPrototype
SB Editor V.1.0 for LandXplorer Studio V.3.x

Element	FloorPrototype	FloorDecoration
SmartBuilding		
Floor 0	Erdgeschoss	unnamed
Floor 1	**Erdgeschoss**	**unnamed**
Floor 2	2	unnamed
Floor 3	2	unnamed
Floor 4	2	unnamed
Floor 5	2	unnamed
Floor 6	3	unnamed
Floor 7	4	unnamed
Floor 8	4	unnamed
Floor 9	5	unnamed
Floor 10	5	unnamed
Floor 11	5	unnamed
Floor 12	5	unnamed
Floor 13	6	unnamed
Floor 14	6	unnamed
Floor 15	7	unnamed
Floor 16	7	unnamed
Floor 17	7	unnamed

FloorPrototype
- Groundplan
  - **Polygon**
- Wall 0

Figure 3: The smart-building editor of the LandXplorer 3D city model system.

projected onto a building regardless of the geometry structure of the building. Conceptually, for each projective texture we define an invisible ghost-wall, which serves as projector wall. Projective textures are effortless to assign for complex building shapes and, therefore, can be used for the rapid modeling of existing buildings.

- Composition of texture patterns: For individual sections of a building façade, we can specify a material texture and a window texture using wall sections. Each wall section belongs to a certain wall object and specifies a range on this wall. Using catalogues of standard materials and window types, wall sections for a façade can be edited instantaneously.

The advantage of projective textures is that they can directly map facade data captured by digital photography. In contrast, the composition of texture patterns frequently models typical, but non-authentic facades. It does not involve the problem of occluded facade parts, e.g., by trees in front of the facade, and allows for ad-hoc modeling of buildings that are only planned or proposed.

3.5 Assigning Application-Specific Data

Floor descriptions provide means to integrate application-specific data into smart buildings. To represent the data, smart buildings use generic two-column, multi-row tables, called attribute tables. An attribute table stores key-value pairs. Both, keys and values, are formatted as strings, which can contain textual, categorical, and numerical contents.

For a single smart building we can specify general attribute tables such as address, owner, usage, etc. for the whole building. In addition, attribute tables can be specified for individual floors and for individual wall sections.

Meta-data sections are a special form of wall sections that specify attribute tables (instead of appearance attributes). In contrast to wall sections for textures, floor attribute tables and wall sections are not defined for floor prototypes but for each floor separately. For instance, in an office building, each floor could be rent by a different company although the appearance of all floors would be equal.

4. CONCLUSIONS

Smart buildings provide a concept for continuous level-of-detail modeling of building models and target at their incremental development. Due to their per-floor concept, they can be perfectly used by direct-manipulation interfaces, providing an intuitive tool for building refinement. Smart buildings aim at the main use case in 3D city model applications, the project-driven and event-driven customization and reengineering of city model components.

The smart building concept has been implemented as a part of the LandXplorer system, an authoring and presentation tool for 3D city models and 3D landscape models. We observed in a variety of use cases that with smart buildings we can approximate complex building models in a time efficient way. Of course, smart buildings are not intended to substitute CAD models but provide a graphics-centered, application-centered modeling schema. They are also suited for large-scale 3D city models, and they can be mapped to an internal graphics representation that allows for real-time photorealistic and non-photorealistic rendering (Döllner et al. 2005).

One application example is a decision-support system in urban planning. Using smart buildings, proposed changes can be interactively performed within the geovirtual 3D environment, so that the effect of the modification can be evaluated and discussed immediately. Another application is concerned with managing interactive 3D location plans. Using smart buildings, building models that exhibit characteristic exterior and interior features are created and maintained by the smart-building editor.

As next steps, we are working on analyzing and mapping arbitrary CityGML-based building models to smart buildings.

We also would like to investigate high-level operations for smart buildings (e.g., adding penthouses, constructing roofs; drilling courtyards, designing entrances, etc.) and using constraints to assist the construction and refinement process - the buildings would become smarter. We also expect the smart building could be a powerful intermediate representation for authoring tools based on CityGML.

ACKNOWLEDGMENTS

We would like to thank our colleagues of the LandXplorer system (www.3dgeo.de), which has been used as implementation platform. We also would like to thank Falko Liecke, Senate of Economy Berlin, and Takis Sgouros, Senate of Urban Development Berlin, for their collaboration within the Berlin 3D City Model project. We would like to thank the German environmental foundation Deutsche Bundesstiftung Umwelt for supporting our work within the Lenné-3D research project (www.lenne3d.de).

REFERENCES

Akenine-Möller, T., Haines, E., 2002. *Real-Time Rendering*. A K Peters Ltd, 2nd Ed.

Altmaier A., Kolbe, T.H., 2003. Applications and Solutions for Interoperable 3D Geo-Visualization. *Proceedings of the Photogrammetric Week 2003*, Wichmann Verlag.

Bauer J., Klaus A., Karner K., Schindler K., Zach C., 2002. MetropoGIS: A Feature based City Modeling System, *Proc. Photogrammetric Computer Vision 2002 (PCV02)* - ISPRS Comission III Symposium, Graz, Austria.

Davis, D., Ribarsky, W., Jiang, T.Y., Faust, N., Ho, S., 1999. Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects. *IEEE Visualization 1999*, 437-440.

Döllner, J., Buchholz, H., Nienhaus, M., Kirsch, K., 2005: Illustrative Visualization of 3D City Models, *Proceedings of SPIE - Visualization and Data Analysis 2005 (VDA)*, San Jose, CA, USA, 42-51.

Felkel, P., Obdrmalek, S. 1998. Straight Skeleton Implementation. *14th Spring Conference on Computer Graphics*, 210-218.

Förstner, W. 1999. 3D City Models: Automatic and Semiautomatic Acquisition Methods. *Proceedings Photogrammetric Week*, University of Stuttgart, 291-303.

Hu, J., You, S., Neumann, U. 2003. Approaches to Large-Scale Urban Modeling. *IEEE Computer Graphics and Applications*, 23(6):62-69.

T. H. Kolbe, G. Gröger, and L. Plümer: CityGML – Interoperable Access to 3D City Models. *First International Symposium on Geo-Information for Disaster Management* GI4DM, 2005.

Laycock, R.G., Day, A.M. Automatically Generating Roof Models from Building Footprints. *Proceedings of WSCG,* Poster Presentation, 2003.

Lindstrom, P., Pascucci, V., 2002. Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization, *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239-254.

Parish, Y., Müller, P. 2001. Procedural Modeling of Cities. *Computer Graphics (Proceedings of SIGGRAPH 2001),* 301-308.

Ribarsky, W., Wasilewski, T., Faust N. 2002. From Urban Terrain Models to Visible Cities. *IEEE Computer Graphics and Applications*, 22(4):10-15.

Schaufler, G. 1998. *Rendering Complex Virtual Environments*. Dissertation, University of Linz.

Willmott, J., Wright, L.I., Arnold, D.B., Day, A.M. 2001. Rendering of Large and Complex Urban Environments for Real-Time Heritage Reconstructions. *Proceedings VAST 2001: The International Symposium on VR, Archaeology, and Cultural Heritage*, 111-120.

Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W. 2003. Instant Architecture. *Computer Graphics (Proceedings of SIGGRAPH)*, 669-677.

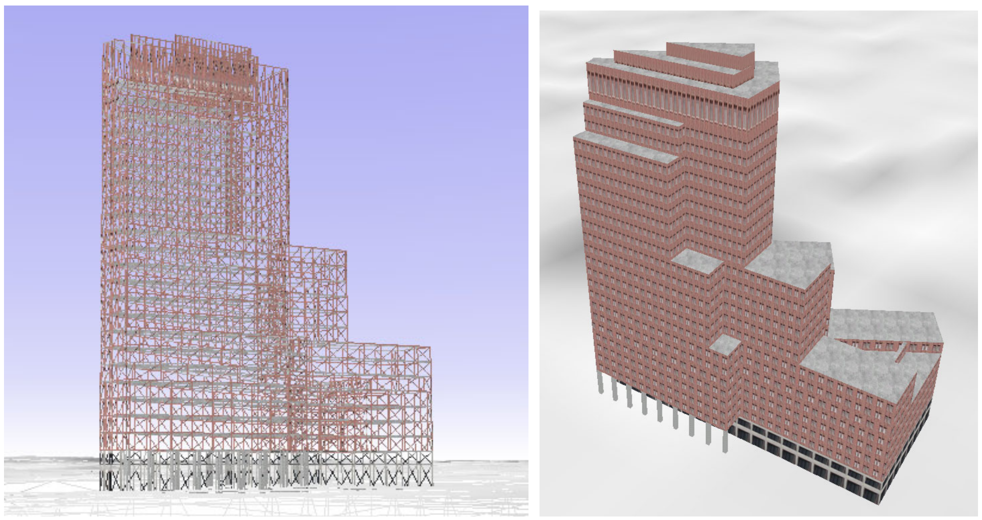


Figure 4: The Kollhoff building at the Potsdamer Platz, represented as smart building for the Berlin 3D city model.